WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Thursday, April 08, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ	
☐	L11	L10 and (new class or custom$ or selective or detect$)	15
☐	L10	L3 and l9	18
☐	L9	L8 and (reloading or reloaded or reload)	63
☐	L8	L5	509
		DB=EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L7	L6 and (new class or custom$ or selective or detect$)	6
☐	L6	L5	23
		DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=ADJ	
☐	L5	L4 and dynamic	532
☐	L4	Class near (loading or loader)	985
		DB=PGPB,USPT,USOC; PLUR=YES; OP=ADJ	
☐	L3	L2 or l1	4813
☐	L2	718/1,100-105.ccls.	3081
☐	L1	717/146-167.ccls.	1796

END OF SEARCH HISTORY

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 18 of 18 returned.

☐ 1. Document ID: US 20040015936 A1

Using default format because multiple data bases are involved.

```
  L10: Entry 1 of 18                 File: PGPB                 Jan 22, 2004

PGPUB-DOCUMENT-NUMBER: 20040015936
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20040015936 A1

TITLE: Dynamic class reloading mechanism

PUBLICATION-DATE: January 22, 2004

INVENTOR-INFORMATION:
 NAME                           CITY        STATE  COUNTRY  RULE-47
 Susarla, Hanumantha Rao        Bangalore   IN     US
 Garg, Mukesh                   Bangalore   IN     US
 E, Sandhya                     Bangalore   IN     US

US-CL-CURRENT: 717/166; 717/165
```

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Ir

☐ 2. Document ID: US 20030033344 A1

```
  L10: Entry 2 of 18                 File: PGPB                 Feb 13, 2003

PGPUB-DOCUMENT-NUMBER: 20030033344
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20030033344 A1

TITLE: Method and apparatus for suspending a software virtual machine

PUBLICATION-DATE: February 13, 2003

INVENTOR-INFORMATION:
 NAME                       CITY             STATE   COUNTRY   RULE-47
 Abbott, Paul Harry         Kings Somborne           GB
 Chapman, Matthew Paul      Eastleigh                GB

US-CL-CURRENT: 718/1
```

ABSTRACT:

A computer system includes a software virtual machine (such as Java) for running one or more applications. An object is provided that is responsive to a call from an application for placing the virtual machine and application into a state of suspension. This involves interrupting all current threads, and recording the state of the components of the virtual machine, including heap, threads, and stack, into a serialization data structure. Subsequently the serialization data structure can be invoked to resume the virtual machine and application from the state of suspension. Note that many virtual machines can be cloned from the single stored data structure. One benefit of this approach is that a new virtual machine can effectively be created in an already initialized state.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Ir

---

☐ 3. Document ID: US 20030005027 A1

L10: Entry 3 of 18 File: PGPB Jan 2, 2003

PGPUB-DOCUMENT-NUMBER: 20030005027
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20030005027 A1

TITLE: Computer system for detecting object updates

PUBLICATION-DATE: January 2, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Borman, Samuel David	Southsea		GB	
Slattery, Edward John	Winchester		GB	

US-CL-CURRENT: 718/104

ABSTRACT:

A computer system is used to run one or more programs. It includes a memory having at least a first heap and a second heap in which objects are stored, with a first object being stored on the first heap. A write barrier is provided for detecting that said the first object has been updated by a program to include a first reference to a memory location in the second heap.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Ir

---

☐ 4. Document ID: US 20020112227 A1

L10: Entry 4 of 18 File: PGPB Aug 15, 2002

PGPUB-DOCUMENT-NUMBER: 20020112227
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020112227 A1

TITLE: Dynamic compiler and method of compiling code to generate dominant path and to handle exceptions

PUBLICATION-DATE: August 15, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Kramskoy, Jeremy Paul	Long Kitton		GB	
Charnell, William Thomas	Great Missenden		GB	
Darnell, Stephen	Flackwell Heath		GB	
Dias, Blaise Abel Alec	Hillingdon		GB	
Guthrie, Philippa Joy	West Park Farm		GB	
Plummer, Wayne	High Wycombe		GB	
Sexton, Jeremy James	High Wycombe		GB	
Wynn, Michael John	Maidenhead		GB	
Rautenbach, Keith	Cardiff		GB	
Thomas, Stephen Paul	High Wycombe		GB	

US-CL-CURRENT: 717/148; 717/139, 717/158, 717/162

ABSTRACT:

A dynamic compiler and method of compiling code to generate a dominate path and handle exceptions. The dynamic compiler includes an execution history recorder that is configured to record the number of times a fragment of code is interpreted. When the code is interpreted a threshold number of times, the code is queued for compilation. The execution history recorder also keeps track of where transfer of control came from and where transfer of control goes to for each fragment of code that is executed, thereby allowing for compilation of a dominant path of code. If the execution of code deviates from the dominant path of compiled code (such as when an exception occurs), a fallback interpreter is utilized to interpret the fragment of code to be executed.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw. Desc | Ir

---

☐ 5. Document ID: US 20020108107 A1

L10: Entry 5 of 18 File: PGPB Aug 8, 2002

PGPUB-DOCUMENT-NUMBER: 20020108107
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020108107 A1

TITLE: Hash table dispatch mechanism for interface methods

PUBLICATION-DATE: August 8, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Darnell, Stephen	Flackwell Heath		GB	
Charnell, William Thomas	Great Missenden		GB	

Plummer, Wayne	High Wycombe	GB
Alec Dias, Blaise Abel	Hillington	GB
Guthrie, Philippa Joy	Leighton Buzzard	GB
Kramskoy, Jeremy Paul	Long Kitton	GB
Sexton, Jeremy James	High Wycombe	GB
Wynn, Michael John	Maidnhead	GB
Rautenbach, Keith	Cardiff	GB
Thomas, Stephen Paul	High Wycombe	GB

US-CL-CURRENT: 717/153; 717/154

ABSTRACT:

A hash table dispatch mechanism for interface Methods. The mechanism reduces dispatch times during the execution of an object-oriented language program. An interface hash table having a pointer as an index for either a specific location in a corresponding dispatch table or an interface Method of the program is created for a dispatch table. The interface hash table has an address and a plurality of slots having a hash value related to an interface Method. The mechanism includes a recovery Method for resolving conflicts when two or more slots in the interface hash table contain clashing values.

☐ 6. Document ID: US 20020108106 A1

L10: Entry 6 of 18 File: PGPB Aug 8, 2002

PGPUB-DOCUMENT-NUMBER: 20020108106
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020108106 A1

TITLE: Method and system of cache management using spatial separation of outliers

PUBLICATION-DATE: August 8, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Kramskoy, Jeremy Paul	Long Kitton		GB	
Charnell, William Thomas	Great Missenden		GB	
Darnell, Stephen	Flackwell Heath		GB	
Alec Dias, Blaise Abel	Hillington		GB	
Guthrie, Philippa Joy	Nr. Leighton Buzzard		GB	
Plummer, Wayne	High Wycombe		GB	
Sexton, Jeremy James	High Wycombe		GB	
Wynn, Michael John	Maidenhead		GB	
Rautenbach, Keith	Cardiff		GB	
Thomas, Stephen Paul	High Wycombe		GB	

US-CL-CURRENT: 717/148; 717/139, 717/158, 717/162

ABSTRACT:

A method and a system of cache management using spatial separation of outliers. The system includes a dynamic compiler arranged to create compiled fragments of code having dominant code blocks and outliers. Memory coupled to the dynamic compiler is managed by a compiler manager such that dominant code blocks are stored in one portion of the memory and the outliers are stored in another portion of the memory. Storing the dominant path code separate from the outliers increases efficiency of the system.

☐ 7. Document ID: US 20020104077 A1

L10: Entry 7 of 18 File: PGPB Aug 1, 2002

PGPUB-DOCUMENT-NUMBER: 20020104077
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020104077 A1

TITLE: Multi-threaded fragment patching

PUBLICATION-DATE: August 1, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Charnell, William Thomas	Great Missenden		GB	
Plummer, Wayne	High Wycombe		GB	
Darnell, Stephen	Flackuellheath		GB	
Dias, Blaise Abel Alec	Hillingdon		GB	
Guthrie, Philippa Joy	Buzzard		GB	
Kramskoy, Jeremy Paul	Long Kitton		GB	
Sexton, Jeremy James	High Wycombe		GB	
Wynn, Michael John	Maidenhead		GB	
Rautenbach, Keith	Cardiff		GB	
Thomas, Stephen Paul	High Wycombe		GB	

US-CL-CURRENT: 717/162; 717/139, 717/140

ABSTRACT:

A method and system of multi-threaded fragment patching. The method provides a link in a self-modifying multi-threaded computer system between a first and a second piece of compiled code where the first piece of compiled code includes a control transfer instruction to the second piece of compiled code. The link is formed by inserting a patch from the first piece of compiled code to the second piece of compiled code. The patch may be a direct reference or a reference to an outlier.

☐ 8. Document ID: US 20020046228 A1

L10: Entry 8 of 18 File: PGPB Apr 18, 2002

PGPUB-DOCUMENT-NUMBER: 20020046228
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020046228 A1

TITLE: Method and system for facilitating access to a lookup service

PUBLICATION-DATE: April 18, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Scheifler, Robert W.	Somerville	MA	US	
wollrath, Ann M.	Groton	MA	US	
Waldo, James H.	Dracut	MA	US	

US-CL-CURRENT: 718/1; 709/220

ABSTRACT:

Methods and systems are provided that facilitate access to a service via a lookup service. A lookup service defines a network's directory of services and stores references to these services. A client desiring use of a service on the network accesses the lookup service, which returns the stub information that facilitates the user's access of the service. The client uses the stub information to access the service.

Full | Title | Citation | Front | Review | Classification | Date | Reference | Sequences | Attachments | Claims | KWIC | Draw Desc | In

---

☐ 9. Document ID: US 20020042807 A1

L10: Entry 9 of 18 File: PGPB Apr 11, 2002

PGPUB-DOCUMENT-NUMBER: 20020042807
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20020042807 A1

TITLE: Low-contention grey object sets for concurrent, marking garbage collection

PUBLICATION-DATE: April 11, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Thomas, Stephen Paul	High Wycombe		GB	
Charnell, William Thomas	Great Missenden		GB	
Darnell, Stephen	Flackwell Heath		GB	
Dias, Blaise Abel Alec	Hillingdon		GB	
Guthrie, Philippa Joy	Leighton Buzzard		GB	
Kramskoy, Jeremy Paul	Long Kitton		GB	
Sexton, Jeremy James	High Wycombe		GB	

Wynn, Michael John	Maidenhead	GB
Rautenbach, Keith	Cardiff	GB
Plummer, Wayne	High Wycombe	GB

US-CL-CURRENT: 718/1

ABSTRACT:

A method and system of carrying out garbage collection in a computer system. Specifically, the method and system utilize low contention grey object sets for concurrent marking garbage collection. A garbage collector traces memory objects and identifies memory objects according to a three-color abstraction, identifying a memory object with a certain color if that memory object itself has been encountered by the garbage collector, but some of the objects to which the memory object refers have not yet been encountered. A packet manager organizes memory objects identified with the certain color into packets, provides services to obtain empty or partially full packets, and obtain full or partially full packets, and verifies whether a packet of the certain color is being accessed by one of the threads of the garbage collector.

☐ 10. Document ID: US 20010051970 A1

L10: Entry 10 of 18 File: PGPB Dec 13, 2001

PGPUB-DOCUMENT-NUMBER: 20010051970
PGPUB-FILING-TYPE: new
DOCUMENT-IDENTIFIER: US 20010051970 A1

TITLE: Virtual machine with reinitialisation

PUBLICATION-DATE: December 13, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	COUNTRY	RULE-47
Webb, Alan Michael	Chandlers Ford		GB	

US-CL-CURRENT: 718/1

ABSTRACT:

A computer system includes a virtual machine supporting an object-oriented environment, in which programs to run on the virtual machine are formed from classes loaded into the virtual machine by a class loader. A class must be initialized before being used by a program. A first application is started on the virtual machine, and a set of one or more classes are loaded and initialized for the first application, which is then run. After the first application has finished running, at least one class from the first application is reset. A second application is then started on the virtual machine. This (re)-initialises any classes that have been reset from the first application, prior to using them, but does not have to reload any reset classes.

☐ 11. Document ID: US 6694346 B1

L10: Entry 11 of 18 File: USPT Feb 17, 2004

US-PAT-NO: 6694346
DOCUMENT-IDENTIFIER: US 6694346 B1

TITLE: Long running, reusable, extendible, virtual machine

DATE-ISSUED: February 17, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Aman; Jeffrey D.	Poughkeepsie	NY		
Bordawekar; Rajesh R.	Yorktown Heights	NY		
Brown; Michael Wayne	Georgetown	TX		
Dillenberger; Donna Ngar-Ting	Yorktown Heights	NY		
Emmes; David B.	Poughkeepsie	NY		
Schmidt; Donald William	Stone Ridge	NY		
Sehorne; Mark Alvin	Round Rock	TX		

US-CL-CURRENT: 718/104; 718/1

ABSTRACT:

In a virtual machine environment, the invention enables creation of a long running, reusable, virtual machine are disclosed. The environment includes a shared heap where requisite runtime code to bring the virtual machine into a `ready` mode are loaded, linked, verified, initialized and compiled. Subsequent virtual machines are started and jointly use the shared heap. Applications create their objects in `private heaps` that are exclusively reserved for the respective applications. At the end of execution of an application, each private heap is reinitialized. Static initializers are run in a persistent area of each private heap. This persistent area is reset to its initial values in between execution of applications. This obviates the need to terminate the virtual machine.

24 Claims, 7 Drawing figures
Exemplary Claim Number: 20
Number of Drawing Sheets: 6

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Ir

☐ 12. Document ID: US 6675381 B1

L10: Entry 12 of 18 File: USPT Jan 6, 2004

US-PAT-NO: 6675381
DOCUMENT-IDENTIFIER: US 6675381 B1

TITLE: Software-module dynamic loader, a software-module dynamic loading method and a medium storing the software-module dynamic loading method

DATE-ISSUED: January 6, 2004

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Yamaguchi; Tomohisa	Tokyo			JP

US-CL-CURRENT: 717/168; 717/166

ABSTRACT:

As a program configuration, a step of requesting a basic side to change an application class as an application module and a core class for changing the application class to a new version by themselves is provided in them. In a platform side for loading each of programs, a sub-core class loader for loading a new version based on a request for changing is provided. In a program of the base class in which the sub-core class loader is provided, a step of starting an operation of the application after loading the new version is provided.

7 Claims, 10 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 8

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw. Desc | In

☐ 13. Document ID: US 6671707 B1

L10: Entry 13 of 18 File: USPT Dec 30, 2003

US-PAT-NO: 6671707
DOCUMENT-IDENTIFIER: US 6671707 B1

TITLE: Method for practical concurrent copying garbage collection offering minimal thread block times

DATE-ISSUED: December 30, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Hudson; Richard L.	Northamptom	MA		
Moss; J. Eliot B.	Amherst	MA		

US-CL-CURRENT: 707/206; 718/100

ABSTRACT:

A method for practical concurrent copying garbage collection offering minimal thread blocking times. The method comprises achieving dynamic consistency between objects in an old memory space and objects in a new memory space. Threads are allowed to progress during garbage collection and threads are flipped one at a time. No read barrier is

required.

29 Claims, 14 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 10

☐ 14. Document ID: US 6654793 B1

L10: Entry 14 of 18 File: USPT Nov 25, 2003

US-PAT-NO: 6654793
DOCUMENT-IDENTIFIER: US 6654793 B1

TITLE: System and method for facilitating dynamic loading of stub information to enable a program operating in one address space to invoke processing of a remote method or procedure in another address space

DATE-ISSUED: November 25, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Wollrath; Ann M.	Groton	MA		
Waldo; James H.	Dracut	MA		
Riggs; Roger	Burlington	MA		

US-CL-CURRENT: 709/217; 717/165, 719/330, 719/332

ABSTRACT:

A stub retrieval and loading subsystem is disclosed for use in connection with a remote method invocation system. The stub retrieval and loading subsystem controls the retrieval and loading of a stub for a remote method, into an execution environment, to facilitate invocation of the remote method by a program executing in the execution environment. The stub retrieval subsystem includes a stub retriever for initiating a retrieval of the stub and stub loader for, when the stub is received by the stub retriever, loading the stub into the execution environment, thereby to make the stub available for use in remote invocation of the remote method. In one embodiment, the stub retrieval and loading subsystem effects the retrieval and loading for a program operating in one address space provided by one computer, of stub class instances to effect the remote invocation of methods which are provided by objects operating in another address space, which may be provided by the same computer or a different computer.

20 Claims, 6 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 6

☐ 15. Document ID: US 6625803 B1

L10: Entry 15 of 18 File: USPT Sep 23, 2003

US-PAT-NO: 6625803
DOCUMENT-IDENTIFIER: US 6625803 B1

TITLE: Automatic generation of text and computer-executable code for run-time use

DATE-ISSUED: September 23, 2003

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Massena; Jay Loren	Seattle	WA		
Hodges; C. Douglas	Redmond	WA		

US-CL-CURRENT: 717/100; 717/115, 717/167

ABSTRACT:

The present invention provides a method, apparatus, and medium for adding text and text-based components to a Web page hosted on a server. A control, which is run at the designing time of the web page (design-time), when implemented, writes HTML information to a created web page. The created HTML information may include text and other text based components (client and server scripting, applets, ActiveX controls, JAVA scripting, and other components). Through the use of OLF, the controls incorporate author-friendly capabilities including in-place editing, property sheets, and persistence. Through the use of these controls, authors may automate the web page generation process and eliminate redundant coding.

26 Claims, 5 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 5

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | Ir

---

☐ 16. Document ID: US 6339841 B1

L10: Entry 16 of 18 File: USPT Jan 15, 2002

US-PAT-NO: 6339841
DOCUMENT-IDENTIFIER: US 6339841 B1

TITLE: Class loading model

DATE-ISSUED: January 15, 2002

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Merrick; Roland Albert	Harvington			GB
Webb; Alan Michael	Chandlers Ford			GB

US-CL-CURRENT: 717/166; 707/103R, 709/203, 717/118

ABSTRACT:

This invention relates to a method of loading Java ClassFiles on to a Java Virtual Machine. On a regular JVM the ClassFile are loaded as and when required. In this specification there is described a method of implementing an object oriented program language such as Java on a computer. The method comprises identifying a class, one of the basic building blocks of the language, which is not within the program domain, that is not loaded into the Java a Virtual Machine. Next it introduces to the program domain only the minimum components of the class which are necessary for commencing processing of the class. The class may comprise several blocks of data representing the methods of the class, since the class may only have been identified because one of the methods within the class was referenced then only the block of data representing this method is loaded into the Java Virtual Machine along with the other essential components of the class. Other blocks of data representing methods can be loaded as and when required by the programming domain. Redundant method components may be removed from the program domain to save memory.

22 Claims, 4 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 4

☐ 17. Document ID: US 6202208 B1

L10: Entry 17 of 18 File: USPT Mar 13, 2001

US-PAT-NO: 6202208
DOCUMENT-IDENTIFIER: US 6202208 B1
**** See image for Certificate of Correction ****

TITLE: Patching environment for modifying a Java virtual machine and method

DATE-ISSUED: March 13, 2001

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Holiday, Jr.; Matthew R.	Allen	TX		

US-CL-CURRENT: 717/166; 707/103R, 711/6, 717/168

ABSTRACT:

The invention includes a patch environment for a modifying a program executed by a Java Virtual Machine ("JVM") while the program is being executed. The patch environment has a patch data structure defined on an electronic memory of the computer. The patch data structure has at least one Java patch for modifying a loader environment of the JVM. A plurality of data items contained in a data structure defined on the electronic memory of the computer represents each patch of the patch data structure. A second data item is contained in a second data structure defined on the electronic memory of the computer, the data item representing each applied patch of the patch data structure that modifies the loader environment of the JVM. The method of the present invention applies an ordered set of changes to a Java program while running under the control of a Java Virtual Machine having a loader environment which manages the program's loaded software. A patch environment is created such that the patch environment can alter the loader environment of the JVM. A patch file is generated containing a change to be applied to the loaded software and is loaded into the patch environment. The patch is then applied to the loaded software by changing the loader environment of the JVM while the Java program is running.

12 Claims, 6 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 6

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | In

---

☐ 18. Document ID: US 6035119 A

L10: Entry 18 of 18 File: USPT Mar 7, 2000

US-PAT-NO: 6035119
DOCUMENT-IDENTIFIER: US 6035119 A
**** See image for Certificate of Correction ****

TITLE: Method and apparatus for automatic generation of text and computer-executable code

DATE-ISSUED: March 7, 2000

INVENTOR-INFORMATION:

NAME	CITY	STATE	ZIP CODE	COUNTRY
Massena; Jay Loren	Seattle	WA		
Hodges; C. Douglas	Redmond	WA		

US-CL-CURRENT: 717/100; 717/115; 717/167

ABSTRACT:

The present invention provides a method, apparatus, and medium for adding text and text-based components to a Web page hosted on a server. A control, which is run at the designing time of the web page (design-time), when implemented, writes HTML information to a created web page. The created HTML information may include text and other text based components (client and server scripting, applets, ActiveX controls, Java scripting, and other components). Through the use of OLE, the controls incorporate author-friendly capabilities including in-place editing, property sheets, and persistence. Through the use of these controls, authors may automate the web page generation process and eliminate redundant coding.

25 Claims, 5 Drawing figures
Exemplary Claim Number: 1
Number of Drawing Sheets: 5

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | In

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Terms	Documents
L3 and L9	18

Display Format: - Change Format

Previous Page   Next Page   Go to Doc#

Hit List

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Search Results - Record(s) 1 through 6 of 6 returned.

☐ 1. Document ID: NN9611107

L7: Entry 1 of 6 File: TDBD Nov 1, 1996

TDB-ACC-NO: NN9611107

DISCLOSURE TITLE: Java Dynamic Class Loader

PUBLICATION-DATA:
IBM Technical Disclosure Bulletin, November 1996, US

VOLUME NUMBER: 39
ISSUE NUMBER: 11
PAGE NUMBER: 107 - 108

DISCLOSURE TEXT:

Java Dynamic Class Loader Java is a portable, interpreted, high-performance, simple, object-oriented programming language environment developed at Sun Microsystems. The next generation of network centric applications needs to download code from either the network or the local disk. Java offers facilities for transmitting applications, called applets, over the network and running them on multiple client machines (1,2). Java standard class loader is responsible for loading classes from either the network or the local disk. Once a program instantiates a new class, the class loader fetches the class from the local cache, in case the class has been loaded already, or loads it explicitly. This mechanism is quite efficient because it avoids reloading the same class multiple times. The main drawback is related to the inability of the class loader to detect whether a class has been modified and then load the new version of it. This prevents updating applications written in Java while running. Java does not allow substitution of the standard class loader with a custom class loader, but it allows loading selected classes using a custom class loader. This means that the application is loaded using the standard class loader whereas the user can load selected classes using a custom class loader. Writing a custom class loader that has the ability to detect whether a class is changed and then loading the new class version does not solve the problem. The Java byte code verifier is a component of the Java runtime that checks the byte code, i.e., the class being loaded, and decides whether the class can be instantiated. If a class is to be reloaded, the byte code verifier detects that the implementation of the class, or of some parent classes, has changed and then it refuses to instantiate a new class instance using the new version of the class. The fact that the current class implementation does not match the previous class implementation is interpreted as a security violation. The Dynamic Class Loader (DCL) herein described allows dynamic creation of object instances of classes whose implementation changes during program execution. Being the standard class loader used by default by Java to load the application, the DCL can be used to load additional classes which are not loaded by the standard class loader. In order to do this, each class loaded by the DCL must be derived by a common superclass or interface. The application knows only this superclass and, therefore, it uses the standard class loader to load all the instances. The subclasses will then be loaded by the DCL: this mechanism works since the superclass, which is the only object class/interface known to the application, will not change during the program execution whereas subclasses can change since they have been loaded by the DCL; hence, they are out of control of the Java class loader. In other words, this solution allows the implementation of a class to be changed while its interface is preserved. For instance, suppose having a drawing application, the application knows only about the class DrawingTool and it uses the DCL to load subclasses of the class DrawingTool such as CircleTool and OvalTool. The standard class loader caches internally the classes it knows up to the DrawingTool class in the class inheritance hierarchy. This means that the byte code verifier will accept new class definitions if and only if the classes loaded by the standard class loader are not modified. Hence, the DCL loads classes derived from the DrawingTool class and uses the standard class loader to resolve the classes above it (i.e., load the superclasses). The DCL allows efficiently loading classes and detects when a new class has been changed. Because the main program knows only about the class DrawingTool, the DCL is responsible for loading the classes CircleTool and OvalTool whereas the classes up to DrawingTool (for instance java.lang.System, java.io.PrintStream and DrawingTool itself) are loaded using the standard class loader using the method findSystemClass of the class ClassLoader. References (1) J. Gosling, H. McGilton, The Java Language Environment, A White Paper Sun Microsystems, Inc. (May 1995). (2) Sun Microsystems, Inc., The Java Virtual Machine Specification Release 1.0 Beta (August 1995).

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc

☐ 2. Document ID: US 20040015936 A1

L7: Entry 2 of 6 File: DWPI Jan 22, 2004

DERWENT-ACC-NO: 2004-132126
DERWENT-WEEK: 200413
COPYRIGHT 2004 DERWENT INFORMATION LTD

TITLE: Dynamic class reloading method for computer system, involves replacing class loader in hierarchy, based on detected class change, during application execution

INVENTOR: E S_GARG, M; SUSARLA, H R ; ,

PRIORITY-DATA: 2001US-292906P (May 22, 2001), 2001US-0895287 (June 29, 2001)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
US 20040015936 A1	January 22, 2004		021	G06F009/45

INT-CL (IPC): G06 F 9/45

ABSTRACTED-PUB-NO: US20040015936A
BASIC-ABSTRACT:

NOVELTY - The classes in application, are loaded by corresponding class loaders (200,202,204,206,208). A class loader in the hierarchy, is replaced by another class loader, for reloading respective class based on detected class change during execution of application.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

(1) software system; and

(2) carrier medium for storing dynamic class reloading program.

USE - For computer system used for providing dynamic class reloading in applications such as Java Beans (EJB), Java 2 platform, Enterprise Edition (J2EE).

ADVANTAGE - Enables reloading dynamic class, reliably.

DESCRIPTION OF DRAWING(S) - The figure shows the block diagram of the class loader stack.

class loaders 200,202,204,206,208

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | C

☐ 3. Document ID: US 20030200350 A1

L7: Entry 3 of 6 File: DWPI Oct 23, 2003

DERWENT-ACC-NO: 2003-832276
DERWENT-WEEK: 200377
COPYRIGHT 2004 DERWENT INFORMATION LTD

TITLE: Class loading/reloading system for computer programming language e.g. Java, provides new class loader for class group that contains changed class during application execution

INVENTOR: KUMAR, A; SUSARLA, H R

PRIORITY-DATA: 2002US-0125949 (April 19, 2002)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
US 20030200350 A1	October 23, 2003		036	G06F009/44

INT-CL (IPC): G06 F 9/44

ABSTRACTED-PUB-NO: US20030200350A
BASIC-ABSTRACT:

NOVELTY - A memory stores program instructions that implement an application which is executable by processor. The application comprises of inter dependent and non-interdependent classes with associated hierarchical stack of class loaders. Each class loader loads the class group of the classes in the application. A new class loader is assigned for a class that changed during execution of application and it loads associated class group.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

(1) method for dynamically reloading classes in application execution;

(2) method for assigning classes to class loaders for application; and

(3) software instructions for implementing new class loader for changed class in application execution.

USE - For providing class dependency graph based class loading and reloading in programming languages such as JAVA, C, C++ using JAVA virtual machine.

ADVANTAGE - The hot-swapping of programmatic logic such as classes, applets and beans in Java programming language is efficiently performed. Hence a domain-independent, flexible, platform independent and robust names pace specification technique is obtained.

DESCRIPTION OF DRAWING(S) - The figure shows the flowchart explaining the dynamic class reloading in application execution.

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | C

☐ 4. Document ID: KR 359640 B, KR 2001056969 A

L7: Entry 4 of 6 File: DWPI Nov 4, 2002

DERWENT-ACC-NO: 2002-047726
DERWENT-WEEK: 200329

TITLE: Class dynamic loading system and operation method thereof

INVENTOR: PARK, Y E

PRIORITY-DATA: 1999KR-0058678 (December 17, 1999)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
KR 359640 B	November 4, 2002		000	H04L012/24
KR 2001056969 A	July 4, 2001		001	H04L012/24

INT-CL (IPC): H04 L 12/24

ABSTRACTED-PUB-NO: KR2001056969A
BASIC-ABSTRACT:

NOVELTY - A class dynamic loading system and an operation method thereof are provided to implement an easier and automatic dynamic loading of a new class based on one time command in a local region system as well as a remote system.

DETAILED DESCRIPTION - A DCL(Dynamic Class Logic) invoker(36) receives information from a developer needed for a class dynamic loading and requests a dynamic loading of a class to the DCL trigger(37). A DCL trigger(37) receives a dynamic loading request of the class from the DCL invoker(36) and outputs a dynamic loading command of the class to an agent(40). The agent(40) includes a network management system core(35), a DCL storing unit(38), and a DCL class list file(39). The network management system core(35) includes

a communication module(31) for a communication with the developer, a kernel(32) for processing the management operation request received from the developer, a tree(33) for implementing a relationship between the management objects, and a management object frame work(34) for providing data and code related to the management objects.

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | C

☐ 5. Document ID: US 6633892 B1, CA 2255035 A1, CA 2255035 C

L7: Entry 5 of 6 File: DWPI Oct 14, 2003

DERWENT-ACC-NO: 2000-638763
DERWENT-WEEK: 200368

TITLE: Archiving tool for archiving files in an archive file that provides customized entry names for the archived files

INVENTOR: CHAN, V S; CHIANG, S S ; STOKES, D K ; THEIVENDRA, L W

PRIORITY-DATA: 1998CA-2255035 (November 30, 1998)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
US 6633892 B1	October 14, 2003		000	G06F012/00
CA 2255035 A1	May 30, 2000	E	025	G06F017/30
CA 2255035 C	January 29, 2002	E	000	G06F017/30

INT-CL (IPC): G06 F 12/00; G06 F 17/00; G06 F 17/30

ABSTRACTED-PUB-NO: CA 2255035A
BASIC-ABSTRACT:

NOVELTY - A dynamic class loader (40) is used in conjunction with a default class loader (30) to load a class into a memory (28) in a form suitable for interpretation by a Java interpreter (26). The class loader maintains a set of pointers to classes that have already been loaded into the memory and the pointers are preferably stored in a hash table and are indexed by class name. The class loader also works in conjunction with one or more byte representations of class files (42) provided by the user or application and a second hash table is used by the class loader to store pointers to these byte representations, which are indexed by class name.

DETAILED DESCRIPTION - AN INDEPENDENT CLAIM is included for a method of archiving files.

USE - Archiving files in a system providing customized entry names.

ADVANTAGE - Enhanced flexibility of use and design of programs.

DESCRIPTION OF DRAWING(S) - The drawing is a schematic diagram of the present invention

Dynamic class loader 40

Default class loader 30

Memory 28

Java interpreter 26

Class files 42

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | C

☐ 6. Document ID: US 6067577 A

L7: Entry 6 of 6 File: DWPI May 23, 2000

DERWENT-ACC-NO: 2000-541990
DERWENT-WEEK: 200049

TITLE: Dynamic binding native method for interaction of computer programs to perform specific computer operations

INVENTOR: BEARD, P C

PRIORITY-DATA: 1996US-0723034 (September 30, 1996)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE	PAGES	MAIN-IPC
US 6067577 A	May 23, 2000		013	G06F009/445

INT-CL (IPC): G06 F 9/445

ABSTRACTED-PUB-NO: US 6067577A
BASIC-ABSTRACT:

NOVELTY - A mechanism in the program language is defined. A class of objects is loaded into the memory of the computer during program execution. The association between the mechanism and a shared library and the class of objects is detected during the loading of the class of objects. An implementation of a method is loaded from the shared library into the memory during the loading of the class of objects.

DETAILED DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

(a) a system for binding functions provided by an operating system to objects in a program;

(b) and a computer-readable storage medium.

USE - For interaction of computer programs to perform specific computer operations.

ADVANTAGE - Provides procedure by which an interpreted byte code program can directly access functions which reside outside of program without having to use intermediate glue code. Provides mechanism via which functions in a non-object-oriented environment, which lie outside of recognized classes, can be utilized by an object-oriented program, and vice versa.

DESCRIPTION OF DRAWING(S) - The figure is the flowchart of the procedure for loading a native library.

Full | Title | Citation | Front | Review | Classification | Date | Reference | | | Claims | KWIC | Draw Desc | C

Clear | Generate Collection | Print | Fwd Refs | Bkwd Refs | Generate OACS

Terms	Documents
L6 and (new class or custom$ or selective or detect$)	6

Display Format: REV | Change Format

Previous Page   Next Page   Go to Doc#